The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

IMAGE DATA COMPRESSION

DRAFT RECOMMENDED STANDARD

CCSDS 122.0-R-2

RED BOOK
July 2005

AUTHORITY

Issue:	Red Book, Issue 2
Date:	July 2005
Location:	Not Applicable

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS Recommended Standards is detailed in *Procedures Manual for the Consultative Committee for Space Data Systems*, and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommended Standard is published and maintained by:

CCSDS Secretariat
Office of Space Communication (Code M-3)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this Recommended Standard is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.

- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:

  - The **standard** itself.

  - The anticipated date of initial operational capability.

  - The anticipated duration of operational service.

- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING FOREWORD:)

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in the *Procedures Manual for the Consultative Committee for Space Data Systems*. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (Roskosmos)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Program Office (NSPO)/Taipei.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Recommended Standard. Its ‘Red Book’ status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document’s technical content.

DOCUMENT CONTROL

Document	Title	Date	Status/Remarks
CCSDS 122.0-R-1	Image Data Compression Draft Issue 1	August 2004	Original Issue, superseded
CCSDS 122.0-R-2	Image Data Compression Draft Issue 2	July 2005	Current Draft Issue

CONTENTS

CONTENTS (continued)

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to establish a Recommended Standard for a data-compression algorithm applied to digital data from payload instruments and to specify how these compressed data shall be inserted into source packets for retrieval and decoding.

Source coding for data compression is a method utilized in data systems to reduce the volume of digital data to achieve benefits in areas including, but not limited to,

a) reduction of transmission channel bandwidth;

b) reduction of the buffering and storage requirement;

c) reduction of data-transmission time at a given rate.

1.2 SCOPE

The characteristics of instrument data are specified only to the extent necessary to ensure multi-mission support capabilities. The specification does not attempt to quantify the relative bandwidth reduction, the merits of each approach discussed, or the design requirements for coders and associated decoders. Some performance information is included in reference [B2].

This Recommended Standard addresses image data compression, which is applicable to a wide range of space-borne digital data, where the requirement is for a scalable data reduction, including the option to use lossy compression, which allows some loss of fidelity in the process of data compression and decompression. Reference [B2] gives an outline for an implementation.

1.3 APPLICABILITY

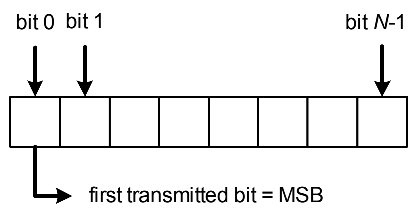
This Recommended Standard applies to data compression applications of space missions anticipating packetized telemetry cross support. In addition, it serves as a guideline for the development of compatible CCSDS Agency standards in this field, based on good engineering practice.

1.4 RATIONALE

The concept and rationale for the Image Data Compression algorithm described herein may be found in reference [B2].

1.5 CONVENTION

In this document, the following convention is used to identify each bit in an N-bit word. The first bit in the word to be transmitted (i.e., the most left justified when drawing a figure) is defined to be 'Bit 0'; the following bit is defined to be 'Bit 1' and so on up to 'Bit N-1'. When the word is used to express an unsigned binary value (such as a counter), the Most Significant Bit (MSB) shall correspond to the highest power of two, i.e., 2^{N-1}.



In accordance with modern data communications practice, spacecraft data words are often grouped into eight-bit 'words' that conform to the above convention. Throughout this Recommended Standard, the following nomenclature is used to describe this grouping:

8-Bit Word = 'Byte'

1.6 DEFINITION

In this document, for any real number x, the largest integer n such that $n \le x$ shall be denoted by

$$n = \lfloor x \rfloor,$$

and correspondingly, the smallest integer n such that $n \ge x$ by

$$n = \lceil x \rceil.$$

The modulus m of a number M with respect to a divisor n is denoted by

$$m = M \bmod n.$$

The statement that a value M is coded mod(n) means the number

$$m = M \bmod n$$

is coded instead of M.

1.7 NOMENCLATURE

The following conventions apply throughout this Recommended Standard:

a) the words ‘shall’ and ‘must’ imply a binding and verifiable specification;

b) the word ‘should’ implies an optional, but desirable, specification;

c) the word ‘may’ implies an optional specification;

d) the words ‘is’, ‘are’, and ‘will’ imply statements of fact.

In the normative sections of this document (sections 3 and 4), informative (i.e., non-normative) text is differentiated from normative text through the following convention:

a) notes introduce brief informative statements;

b) the following headings introduce one or more paragraphs of informative text:

  – Overview;

  – Background;

  – Rationale;

  – Discussion.

1.8 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommended Standard. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommended Standard are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommended Standards.

[1] *Lossless Data Compression*. Recommendation for Space Data Systems Standards. CCSDS 121.0-B-1. Blue Book. Issue 1. Washington D.C.: CCSDS, May 1997.

[2] *Space Packet Protocol*. Recommendation for Space Data Systems Standards. CCSDS 133.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2003.

NOTE – Informative references are listed in annex B.

2 OVERVIEW

2.1 GENERAL

This Recommended Standard defines a particular payload image data compression algorithm that has widespread applicability to many types of instruments. This Recommended Standard does not attempt to explain the theory underlying the operation of the algorithm, which is partially addressed in [B2].

There are two classes of data compression methods: lossless and lossy. Under lossless compression, the original data can be reproduced exactly, while under lossy compression, quantization or other approximations used in the compression process result in the inability to reproduce the original data set without some distortion. The higher fidelity required by lossless compression results in a higher volume of compressed data for a given source data set.

The compression technique described in this Recommended Standard can be used to produce both lossy and lossless compression. An alternative lossless compression technique, which has lower complexity but is not specifically tailored for imagery, has been previously adopted by CCSDS (see reference [1]). The JPEG2000 standard is another image compressor (see reference [B3]). The present Recommended Standard differs from the JPEG2000 standard in several respects:

a) it specifically targets use aboard spacecraft;

b) a careful trade-off has been performed between compression performance and complexity;

c) being less complex, it can be fully implemented in either hardware or software;

d) it has a limited set of options, supporting its successful application without in-depth algorithm knowledge.

The compressor consists of two functional parts, depicted in figure 2-1, a Discrete Wavelet Transform module that performs decorrelation, described in section 3, and a Bit-Plane Encoder which encodes the decorrelated data, described in section 4.

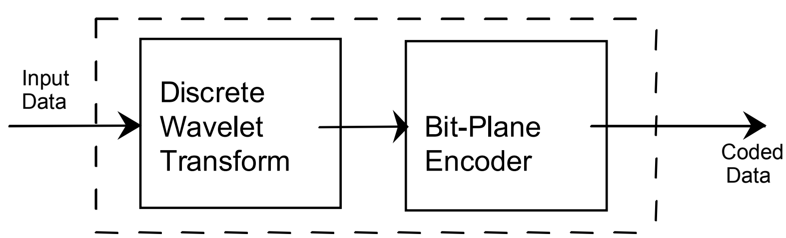


Figure 2-1: General Schematic of the Coder

This Recommended Standard supports both frame-based formats produced, for example, by CCD arrays (called image frames) or strip-based input formats produced by push-broom type sensors (called stripmap images). An image pixel resolution of up to 16 bits is supported. The algorithm specified supports a memory-effective implementation of the compression procedure which does not require large intermediate frames for buffering (see reference [B2]).

2.2 DATA DELIVERY

The encoded bit stream of a frame or a stripmap contains a fixed- or variable-length data field, consisting of a header and coded segment. Values of algorithm parameters used in encoding can be indicated using headers described in section 4. The encoded bit stream with the compression header information shall be delivered reliably in sequence to provide correct data reconstruction and to avoid potential decompression error propagation (see reference [B2]). The transport mechanism for delivery of the encoded bit stream shall support, in the event of bit error, the ability to relocate the header of the next segment.

3 DESCRIPTION OF THE DISCRETE WAVELET TRANSFORM

3.1 OVERVIEW

This Recommended Standard for the decorrelation module makes use of a three-level, two-dimensional (2-d), separable Discrete Wavelet Transform (DWT) with nine and seven taps for low- and high-pass filters, respectively. Such a transform is produced by repeated application of a one-dimensional (1-d) DWT described in 3.3, 3.5, and 3.7. Two specific 1-d wavelets are specified with this Recommended Standard: the 9/7 biorthogonal DWT, referred to as '9/7 Float DWT' or simply 'Float DWT', and a non-linear, integer approximation to this transform, referred to as '9/7 Integer DWT' or simply 'Integer DWT'.

The definition of the functional inverse of either DWT is included in 3.4, 3.6, and 3.8.

While the Float DWT generally exhibits superior compression efficiency in the lossy domain, only the Integer DWT supports strictly lossless compression.

Image data is assumed to use R-bit pixels to represent either signed or unsigned integer values, where $R \leq 16$. A corresponding word length of $R+7$ bits is sufficient to support the dynamic range occurring in the transform domain, to be represented using one bit for the sign and the remaining bits for the absolute value of any wavelet coefficient.

In the case of the Float DWT, the computed wavelet coefficients are rounded to the respective nearest integers before applying the Bit Plane Encoder (BPE—see section 4). In the case of the Integer DWT, before applying the BPE, the computed wavelet coefficients are multiplied by weights that are uniform in each subband (see 3.9).

Implementation schemes are not part of this Recommended Standard but are proposed in reference [B2]. They can differ considerably depending on how time enters the process: whether, e.g., the imaging instrument delivers its data linewise or framewise.

3.2 IMAGE FRAME

3.2.1 Definition of both Float DWT and Integer DWT shall specify the dimensions, i.e., row width and column height, of the image frame being transformed.

NOTE – Frame size is relevant because special filters are applied at frame boundaries. This formal consideration of frame size does not imply that an implementation has to provide buffers for complete frames.

3.2.2 Frame size shall be communicated to the decoder using the segment headers defined in 4.2.

3.2.3 In the case of push-broom mode, there is no restriction to the number of rows that constitute one image frame.

3.2.4 The application of either Float or Integer DWT to an image requires that the image dimensions be integer multiples of eight. If the width or height of the original image is a not a multiple of eight, the image shall be 'padded' by appending the minimum number of extra rows and/or columns to produce an image with both width and height divisible by eight:

a) when columns are added for padding,

   1) columns shall be appended at the right edge of the image, i.e., to the edge having the highest pixel index in the horizontal direction,

   2) pixel values of appended columns are recommended to be copies of the value of the right-most image pixel;

b) when rows are added for padding,

   1) rows shall be appended at the bottom edge of the image, i.e., to the edge having the highest pixel index in the vertical direction,

   2) appended rows are recommended to be identical copies of the last row to the image.

3.2.5 Rows and/or columns of data added for padding shall be discarded after decompression.

3.2.6 After adding any required padding and performing the DWT, the wavelet data shall be grouped into blocks of 64 coefficients each (see section 4). Each block shall contain one coefficient of the top-level low-frequency subband LL_3, referred to as the 'DC coefficient'.

3.2.7 Blocks shall be processed by the Bit Plane Encoder consecutively in the raster scan in the order in which their corresponding DC coefficients occur in LL_3: row by row, each row being processed from left to right.

NOTE – Frame size is not relevant for the Bit Plane Encoder, as it is fed only with blocks of wavelet coefficients. The output of the Bit Plane Encoder consists of so-called segments. Each segment contains the code of a user-selected number of consecutive blocks. The beginnings and endings of the segments may or may not coincide with the beginnings and endings of frames: a segment might code part of one frame, a single complete frame, multiple frames, and/or parts thereof.

3.3 ONE-DIMENSIONAL, SINGLE-LEVEL DWT

3.3.1 9/7 FLOAT TRANSFORM

3.3.1.1 Analysis Filter Coefficients

The 9/7 Float DWT uses two sets of analysis filter coefficients (‘taps’):

$$\begin{aligned}&\{h_{-4}, h_{-3}, h_{-2}, h_{-1}, h_0, h_1, h_2, h_3, h_4\}\\&\{g_{-3}, g_{-2}, g_{-1}, g_0, g_1, g_2, g_3\}\end{aligned} \tag{1}$$

The numerical values of the two sets of taps are specified in table 3-1.

Table 3-1: Analysis Filter Coefficients for the 9/7 Filter

Analysis Filter Coefficients		
i	Lowpass Filter h_i	Highpass Filter g_i
0	0.852698679009	-0.788485616406
±1	0.377402855613	0.418092273222
±2	-0.110624404418	0.040689417609
±3	-0.023849465020	-0.064538882629
±4	0.037828455507	

NOTES

1 The theoretical filter coefficients are real, irrational numbers. These numbers have been approximated in the table to within 12 decimals. Any additional accuracy will not have significant impact on the compression performance, since the wavelet coefficients are rounded to the nearest integer before coding. The arithmetic precision to be used for this DWT is a matter of implementation (word length, fixed point vs. floating point) and is not part of this Recommended Standard.

2 The 1-d DWT is defined for an even number of input samples. The image extension procedure described in 3.2 ensures that the number of samples to which the 1-d DWT is applied is even at all levels.

3.3.1.2 Analysis Filter Operations

For N>1, let

$$\{x_0, x_1, x_2, \ldots, x_{2N-1}\} \tag{2}$$

denote a one-dimensional signal consisting of $2N$ samples. The one-dimensional Float DWT is defined by the following pair of *analysis* filter operations:

$$C_j = \sum_{n=-4}^{4} h_n x_{2j+n}; \qquad D_j = \sum_{n=-3}^{3} g_n x_{2j+1+n}; \quad j = 0,1,\ldots,N-1 \tag{3}$$

The outputs C_j, D_j are referred to as the *coefficients* of the wavelet transform. In equation 3 the filter taps are chosen so that C_j and D_j represent low-pass and high-pass outputs, respectively. That is, C_j is a smoothed (low-pass) version of the original signal, while D_j contains high-pass information.

At the beginning (left boundary) of the signal, equation 3 requires signal values with negative indices, and at the end (right boundary) of the signal, it requires signal values with indices exceeding 2N-1. These values are obtained from the following mirror symmetric signal extension requirement:

$$x_m = x_{-m} \quad for\ m < 0; \qquad x_{2N-1+m} = x_{2N-1-m} \quad for\ m > 0 \tag{4}$$

3.3.2 9/7 INTEGER TRANSFORM

NOTE – An alternative transform in this Recommended Standard is a non-linear approximation to a 9/7 DWT. The non-linearity is introduced as the result of round-off operations used for the sake of producing integer outputs from the decorrelation stage. The non-linear transform can be used to achieve lossless compression.

Like the Float DWT defined in equation 3, the present single-level, 1-d Integer DWT shall map a signal vector (equation 2) to two sets of wavelet coefficients, one high-pass set, D_j, and one low-pass set, C_j, in accordance with equations 5 and 6. Special boundary filters are required at either end of the signal, and lead to adapted formulas for j=0, j=N-2, and j=N-1.

Equations 5 and 6 define the integer transform that shall be used with this Recommended Standard. Given input values of x_i, the D_j values in equation 5 shall be computed first and used subsequently to compute C_j values in equation 6.

$$\begin{aligned}
D_0 &= x_1 - \left\lfloor \frac{9}{16}(x_0 + x_2) - \frac{1}{16}(x_2 + x_4) + \frac{1}{2} \right\rfloor \\
D_j &= x_{2j+1} - \left\lfloor \frac{9}{16}(x_{2j} + x_{2j+2}) - \frac{1}{16}(x_{2j-2} + x_{2j+4}) + \frac{1}{2} \right\rfloor \qquad for\ j = 1,\ldots,N-3 \\
D_{N-2} &= x_{2N-3} - \left\lfloor \frac{9}{16}(x_{2N-4} + x_{2N-2}) - \frac{1}{16}(x_{2N-6} + x_{2N-2}) + \frac{1}{2} \right\rfloor \\
D_{N-1} &= x_{2N-1} - \left\lfloor \frac{9}{8}x_{2N-2} - \frac{1}{8}x_{2N-4} + \frac{1}{2} \right\rfloor
\end{aligned} \tag{5}$$

$$C_0 = x_0 - \left\lfloor -\frac{D_0}{2} + \frac{1}{2} \right\rfloor$$
$$C_j = x_{2j} - \left\lfloor -\frac{D_{j-1} + D_j}{4} + \frac{1}{2} \right\rfloor \qquad for\ j = 1,\ldots,N-1 \tag{6}$$

NOTE – Because of the rounding procedure, the recommended transform is, strictly speaking, non-linear and hence not truly a DWT. Nevertheless, a (non-linear) strict inverse transform exists (see 3.4.2).

3.4 INVERSE DWT

3.4.1 INVERSE 9/7 FLOAT TRANSFORM

3.4.1.1 Synthesis Filter Coefficients

The inverse 9/7 Float DWT uses two sets of synthesis filter coefficients:

$$\{q_{-3}, q_{-2}, q_{-1}, q_0, q_1, q_2, q_3\}$$
$$\{p_{-4}, p_{-3}, p_{-2}, p_{-1}, p_0, p_1, p_2, p_3, p_4\} \tag{7}$$

Their numerical values to within 12 decimals are specified in table 3-2.

Table 3-2: Synthesis Filter Coefficients for the 9/7 Filter

Synthesis Filter Coefficients		
i	Lowpass Filter q_i	Highpass Filter p_i
0	0.788485616406	-0.852698679009
±1	0.418092273222	0.377402855613
±2	-0.040689417609	0.110624404418
±3	-0.064538882629	-0.023849465020
±4		-0.037828455507

3.4.1.2 Synthesis Filter Operations

The one-dimensional DWT shall be inverted by means of a pair of *synthesis* filtering operations:

$$\left.\begin{aligned} x_{2j} &= \sum_{n=-1}^{1} q_{2n} C_{j+n} + \sum_{n=-2}^{1} p_{2n+1} D_{j+n} \\ x_{2j+1} &= \sum_{n=-1}^{2} q_{2n-1} C_{j+n} + \sum_{n=-2}^{2} p_{2n} D_{j+n} \end{aligned}\right\} \qquad (j = 0,1,\ldots,N-1) \tag{8}$$

For correct evaluation of equation 8 at the boundaries, the wavelet coefficient signals C_j, D_j must be extended as follows:

$$\begin{aligned} D_m &= D_{-m-1} \quad \textit{for } m<0; \qquad D_{N-1+m} = D_{N-1-m} \quad \textit{for } m>0 \\ C_m &= C_{-m} \quad \textit{for } m<0; \qquad C_{N-1+m} = C_{N-m} \quad \textit{for } m>0 \end{aligned} \tag{9}$$

3.4.2 INVERSE INTEGER TRANSFORM

Like the inverse Float DWT defined in equation 8, the present inverse Integer DWT maps two sets of wavelet coefficients, a low-pass set, C_j, and a high-pass set, D_j, back to a signal vector x_j in accordance with equations 10 and 11. Special boundary filters are required at either end of the data sets, leading to adapted formulas for j=0, j=1, j=2N-3, and j=2N-1.

$$\begin{aligned} x_0 &= C_0 + \left\lfloor -\frac{D_0}{2} + \frac{1}{2} \right\rfloor \\ x_{2j} &= C_j + \left\lfloor -\frac{D_{j-1}+D_j}{4} + \frac{1}{2} \right\rfloor \qquad \textit{for } j = 1,\ldots,N-1 \end{aligned} \tag{10}$$

$$\begin{aligned} x_1 &= D_0 + \left\lfloor \frac{9}{16}(x_0+x_2) - \frac{1}{16}(x_2+x_4) + \frac{1}{2} \right\rfloor \\ x_{2j+1} &= D_j + \left\lfloor \frac{9}{16}(x_{2j}+x_{2j+2}) - \frac{1}{16}(x_{2j-2}+x_{2j+4}) + \frac{1}{2} \right\rfloor \qquad \textit{for } j = 1,\ldots,N-3 \\ x_{2N-3} &= D_{N-2} + \left\lfloor \frac{9}{16}(x_{2N-4}+x_{2N-2}) - \frac{1}{16}(x_{2N-6}+x_{2N-2}) + \frac{1}{2} \right\rfloor \\ x_{2N-1} &= D_{N-1} + \left\lfloor \frac{9}{8} x_{2N-2} - \frac{1}{8} x_{2N-4} + \frac{1}{2} \right\rfloor \end{aligned} \tag{11}$$

In equation 10, x_{2j} values shall be computed first from the DWT coefficients. These reconstructed values shall then be utilized in computing x_{2j+1} values.

3.5 TWO-DIMENSIONAL SINGLE-LEVEL DWT

The decorrelation of an image requires a *two*-dimensional DWT, which is performed by iterated application of the one-dimensional DWT. Viewing the image as a data matrix consisting of rows and columns of signal vectors, a single-level 2-d DWT shall be performed on the image in the following two steps in the following order:

a) the 1-d DWT shall be performed on each image row, producing a horizontally low-pass and a horizontally high-pass filtered intermediate data array, each half as wide as the original image array, as illustrated in figure 3-1(b);

b) the 1-d DWT shall be applied to each column of both intermediate data arrays to produce four subbands as shown in figure 3-1(c).

NOTE – Transposing the order of row and column processing will result in a transposed image when this standard is used for decompressing. For applications that desire a transposition to be performed after the inverse DWT, the header bit in Segment Header Part 4 (4.2.5) can be used to signal this option.

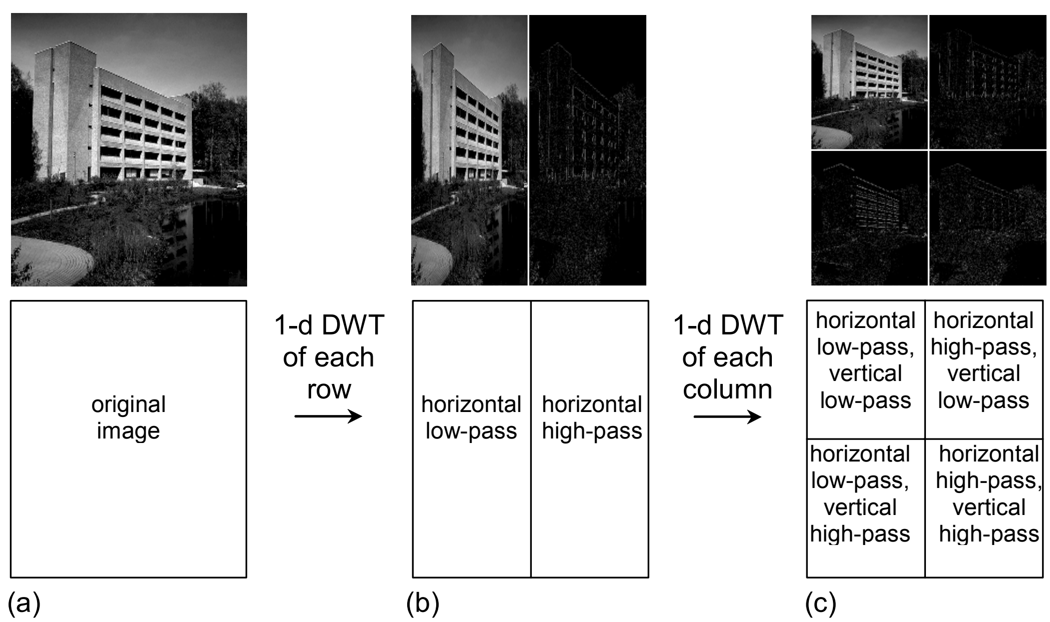


Figure 3-1: 2-d DWT (One Level)

Each of the four subband data arrays obtained is half as wide and half as tall as the original image array. In illustrations, these subbands are often shown arranged as one array which has the same size as the original image array (see figure 3-1(c)). Starting at the upper left and proceeding clockwise in figure 3-1(c), the four subbands are referred to as LL, HL, HH, LH.

3.6 INVERSE OF TWO-DIMENSIONAL DWT

The single-level 2-d DWT transform shall be inverted by repeated application of the 1-d inverse to columns and rows of the transformed data array in the reverse order to that in which the 1-d transforms were applied:

a) each column shall be inverted to produce the intermediate transformed data arrays:

   1) the 1-d DWT inverse shall be applied to columns of the LL and LH subbands to obtain the intermediate horizontal low-pass array of figure 3-1(b),

   2) the 1-d DWT inverse shall be applied to columns of the HL and HH subbands to obtain the intermediate horizontal high-pass array of figure 3-1(b);

b) the 1-d DWT inverse shall be applied to rows of the intermediate horizontal low-pass and horizontal high-pass arrays to recover the original image array.

This inverse procedure shall be used for both lossless and lossy decompression.

3.7 MULTI-LEVEL, TWO-DIMENSIONAL DWT

To increase compression efficiency, correlation remaining in the LL subband after the 2-d DWT decomposition is exploited by applying further levels of DWT decomposition to produce a multi-level 2-d DWT. This produces the pyramidal decomposition described in reference [B4].

This Recommended Standard specifies three levels of decomposition.

At each level, the 2-d DWT described in 3.5 shall be applied to the LL subband produced by the previous level of decomposition.

The image extension procedure described in 3.2 shall be used to assure image dimensions are integer multiples of eight.

NOTE – Figure 3-2 illustrates a three-level 2-d DWT decomposition. At each level of decomposition, the LL subband from the previous level is decomposed, using a 2-d DWT, and is replaced with four new subbands. Each new subband is half the width and half the height of the LL subband from which it was computed. Each additional level of decomposition thus increases the number of subbands by three but leaves unchanged the total number of DWT coefficients used to represent the image data. Following n levels of 2-d DWT decomposition, the total number of subbands is therefore $3n+1$. Following the recommendation of a three-level decomposition, ten subbands are generated. The subbands are typically shown arranged to form an array of the same dimensions as the original image, as is done in figure 3-2. Subscripts are added to LL, HL, HH, and LH to denote the level of decomposition.

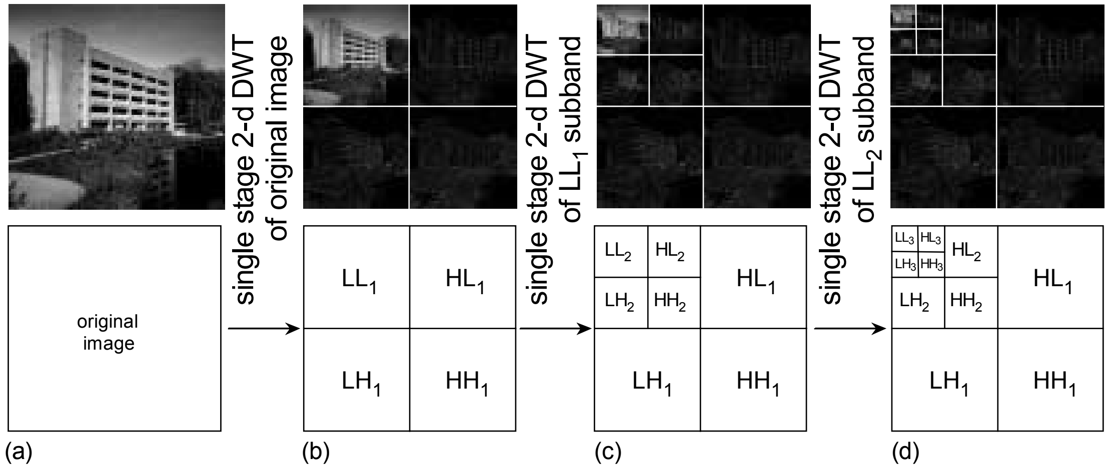


Figure 3-2: Three-Level 2-d DWT Decomposition of an Image

3.8 INVERSE, MULTI-LEVEL 2-D DWT

The inversion process of a multi-level DWT shall be as follows:

a) the four subbands of highest level, LL_3, LH_3, HL_3, HH_3, shall be inverted using an inverse single-level 2-d DWT to yield the single subband LL_2, which then replaces the higher-level subbands in the transform data matrix;

b) the four subbands LL_2, LH_2, HL_2, HH_2 shall be inverted to yield the single subband LL_1, which again replaces the higher-level subbands in the transform data matrix;

c) a final single-level 2-d inverse DWT shall be applied to subbands LL_1, LH_1, HL_1, HH_1 to reproduce the original image.

3.9 SUBBAND WEIGHTS

3.9.1 The Bit-Plane Encoder (BPE) described in section 4 shall be used to encode the subbands produced by the 2-d DWT decomposition.

3.9.2 For the integer transform, all subband coefficients shall be multiplied by their respective weight factors before encoding.

3.9.3 Standard weights for use in multiplying subband coefficients are given in table 3-3.

NOTE – For effective operation, the BPE relies on the same bit plane in each of the subbands having the same relative priority in terms of contribution to overall image distortion. For the integer transform, this is not the case, and the subbands must be scaled.

3.9.4 A user-defined set of weights may be substituted if a user wishes to emphasize data in some subbands over others. This option is flagged in Segment Header Part 4 (4.2.5).

Table 3-3: Standard Subband Weights for 9/7 Integer DWT

Subband	HH_1	HL_1, LH_1	HH_2	HL_2, LH_2	HH_3	HL_3, LH_3	LL_3
Weight factor	2^0	2^1	2^1	2^2	2^2	2^3	2^3

NOTE – The weight factors are chosen to be powers of two to allow scaling to be performed using bit-shift operations. For example, table 3-3 indicates that after applying a three-level 2-d DWT using the integer transform, each of the resulting DWT coefficients in the LH_2 subband should be multiplied by a factor of 2^2, which can be accomplished by two left bit-shift operations.

4 THE BIT PLANE ENCODER (BPE)

4.1 OVERVIEW

Following the DWT, the wavelet coefficients are either rounded to the nearest integer (when the floating-point transform has been used), or scaled using the weighting factors described in 3.9 (when the integer transform has been used). When scaling is performed, one or more of the least-significant bits in many of the subbands are necessarily all zeros. The BPE process takes this into account, i.e., bits that must be all zeros because of the scaling process are not encoded in the BPE. In the description in this section, the coefficient values mentioned refer to values after any scaling or rounding operation. The notation $\mathrm{LS}(\mathrm{HL}_2)$ is used to indicate the number of bits in each coefficient of the HL_2 subband that are necessarily zero as a result of the subband scaling operation. When the 9/7 real DWT is used, the LS values are all zero.

The Bit Plane Encoder (BPE) processes wavelet coefficients in groups of 64 coefficients referred to as a *block*. An example of a block is illustrated in figure 4-1 as comprised of shaded pixels. A block loosely corresponds to a localized region in the original image. Blocks are processed in raster scan order, i.e., rows of blocks are processed from top to bottom, proceeding from left to right horizontally within a row.

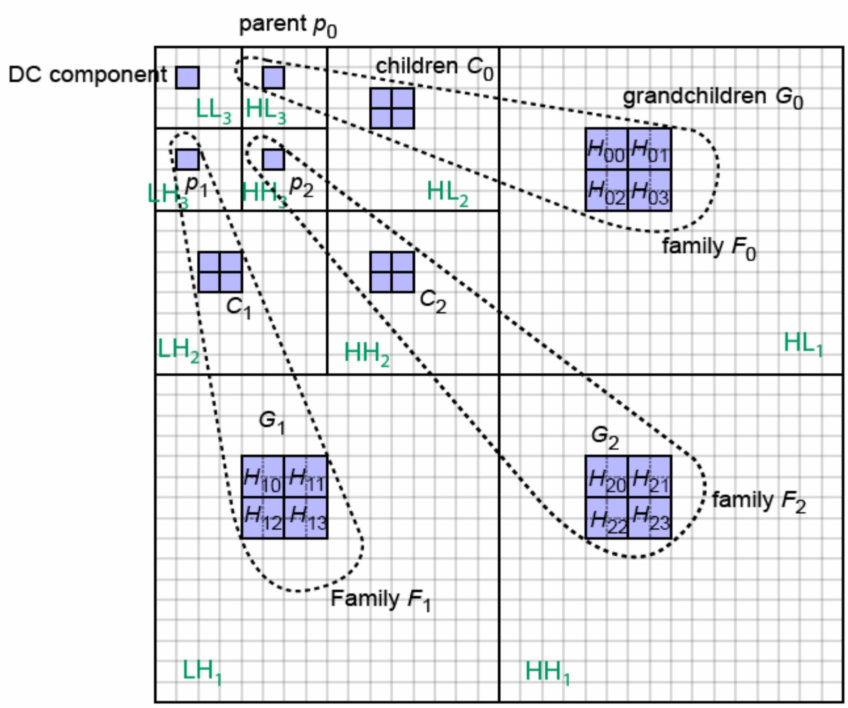


Figure 4-1: Schematic of Wavelet-Transformed Image

Information pertaining to a block of coefficients is jointly encoded by the BPE. A block consists of a single coefficient from the LL_3 subband, referred to as the *DC coefficient*, and

63 *AC coefficients*. The AC coefficients in a block are arranged into three *families*, F_0, F_1 and F_2. Figure 4-1 illustrates a single block of coefficients and the family structure.

Each family F_i in the block has one *parent* coefficient, p_i, a set C_i of four *children* coefficients, and a set G_i of sixteen *grandchildren* coefficients. The grandchildren in family F_i are further partitioned into groups numbered j=0,1,2,3, denoted H_{ij}, as illustrated in figure 4-1. This structure is used for jointly encoding information pertaining to groups of coefficients in the block, as described in 4.5.

A wavelet coefficient is identified by its coordinates within its subband. Thus coordinates (r, c) indicate the wavelet coefficient in row r, column c within the subband, with the upper left pixel in a subband having coordinates (0,0).

The DC coefficient for each block is a single coefficient from the LL_3 subband. The coordinates for the other coefficients in the block can be determined from the coordinates of the DC coefficient. For a block with DC coefficient with coordinates (r, c) within the LL_3 subband, table 4-1 lists the coordinates for the AC coefficients, within their respective subbands of origin. Table 4-2 gives the subband of origin for each type of coefficient, determined by family and family member type. The relationship between all coefficients in a block is further illustrated graphically in figure 4-1.

Table 4-1: Within-Subband Coordinates for Coefficients in a Single Family

Coefficient Group in Family i	Coordinates
Parent, p_i	(r,c)
Children group, C_i	$(2r, 2c)$, $(2r, 2c+1)$, $(2r+1, 2c)$, $(2r+1, 2c+1)$
Grandchildren group, H_{i0}	$(4r, 4c)$, $(4r, 4c+1)$, $(4r+1, 4c)$, $(4r+1, 4c+1)$
Grandchildren group, H_{i1}	$(4r, 4c+2)$, $(4r, 4c+3)$, $(4r+1, 4c+2)$, $(4r+1, 4c+3)$
Grandchildren group, H_{i2}	$(4r+2, 4c)$, $(4r+2, 4c+1)$, $(4r+3, 4c)$, $(4r+3, 4c+1)$
Grandchildren group, H_{i3}	$(4r+2, 4c+2)$, $(4r+2, 4c+3)$, $(4r+3, 4c+2)$, $(4r+3, 4c+3)$

Table 4-2: Subband of Origin for AC Coefficients

	Family 0	Family 1	Family 2
Parent	HL_3	LH_3	HH_3
Children	HL_2	LH_2	HH_2
Grandchildren	HL_1	LH_1	HH_1

A *segment* is defined as a group of S consecutive blocks. Coding of DWT coefficients proceeds segment-by-segment and each segment is coded independently of the others. S can be assigned to any value between 16 and 2^{20} inclusive; the value might be chosen based on the memory available to store the segment. When multiple image frames are transmitted, the coding of each new frame starts with a new segment, i.e., a single segment must not contain code from two separate frames.

DC coefficients are represented using two's-complement representation. Let c_m denote the m^{th} DC coefficient in a segment, i.e., the DC coefficient of the m^{th} block in a segment. The number of bits needed to represent c_m in two's-complement representation is given in equation 12:

$$\begin{array}{ll} 1+\lceil \log_2|c_m| \rceil, & \text{if } c_m < 0 \\ 1+\lceil \log_2(1+c_m) \rceil, & \text{if } c_m \geq 0 \end{array} \tag{12}$$

Within a segment, *BitDepthDC* is defined as the maximum of this value over all DC coefficients (i.e., all values of m) in the segment. Each DC coefficient in the segment is represented using *BitDepthDC* bits, in two's-complement representation.

An AC coefficient is represented using the binary representation of the magnitude of the coefficient, along with a bit indicating the sign when the coefficient is nonzero. $BitDepthAC\_Block_m$ in equation 13 denotes the maximum number of bits needed to specify the magnitude of any AC coefficient in the m^{th} block.

$$BitDepthAC\_Block_m = \lceil \log_2(1 + \max_{i=1,\ldots,63}(|x_{mi}|)) \rceil \tag{13}$$

where x_{mi} denotes the i^{th} AC coefficient in block m.

For each segment, the BPE computes *BitDepthAC*, which denotes the maximum value of $BitDepthAC\_Block_m$ for the segment:

$$\text{BitDepthAC} = \max_{m=0,\ldots,S-1} \text{BitDepthAC\_Block}_m \tag{14}$$

The BPE successively encodes bit planes of coefficient magnitudes in a segment, inserting AC coefficient sign values at appropriate points in the encoded data stream. *Bit plane b* consists of the b^{th} bit of the two's-complement integer representation of each DC coefficient, and the b^{th} bit of the binary integer representation of the magnitude of each AC coefficient. Here, bit plane index b=0 corresponds to the least significant bit. The BPE proceeds from most-significant bit to least significant bit, thus b decreases from one bit plane to the next, beginning with $b = BitDepthAC$-1, and ending with b=0. DWT coefficient resolution effectively improves by a factor of 2 as encoding proceeds from one bit plane to the next. The bit plane coding process is described in 4.5.

Figure 4-2 gives an overview of the structure of a coded segment. Within a segment, header information is encoded. Then quantized DCcoefficients from the blocks are encoded. Then AC bit depths are encoded. Then DWT coefficient blocks are encoded, one bit plane at a time, proceeding from the most significant to the least significant bit plane. The resulting encoded bitstream constitutes an embedded data format that provides progressive transmission.

Segment Header (see §4.2)
Initial coding of quantized DC coefficients (see §4.3)
Coded AC coefficient bit depths (see §4.4)
Coded bit plane b = *BitDepthAC*-1 (see §4.5)
Coded bit plane b = *BitDepthAC*-2 (see §4.5)
…
Coded bit plane b = 0 (see §4.5)

Figure 4-2: Overview of the Structure of a Coded Segment

The tradeoff between reconstructed image quality and compressed data volume for each segment is controlled by specifying the maximum number of bytes in each compressed segment, SegByteLimit, and a 'quality' limit. The quality limit constrains the amount of DWT coefficient information to be encoded, and is specified as a bit plane index and a stopping point within that bit plane, as described in 4.2. Compressed output for a segment is produced until the byte limit or quality limit is reached, whichever comes first.

The encoded bitstream for a segment can be further truncated (or, equivalently, coding can be terminated early) at any point to further reduce the data rate, at the price of reduced image quality for the corresponding segment.

The remainder of this section describes each component of the coded segment. Subsection 4.2 describes the segment header. Subsection 4.3 describes the initial coding of DC coefficients. Subsection 4.4 describes the coded sequence of $BitDepthAC\_Block_m$ values. Subsection 4.5 describes the coding process for a bit plane of the segment.

4.2 SEGMENT HEADER

4.2.1 GENERAL

4.2.1.1 Each compressed segment shall begin with a segment header consisting of the following parts in the following order:

a) Part 1 (three or four bytes, mandatory—see 4.2.2);

b) Part 2 (five bytes, optional—see 4.2.3);

c) Part 3 (three bytes, optional—see 4.2.4);

d) Part 4 (eight bytes, optional—see 4.2.5).

4.2.1.2 By CCSDS convention, all reserved bits in each header part shall be set to 'zero'.

4.2.1.3 Optional header parts may be omitted when the parameters described in a part can be determined without that part, e.g., when those parameters are set to known fixed values for an entire mission.

NOTE – Figure 4-3 gives an overview of the header structure and table 4-3 provides a high-level description of the functionality of each part of the header.

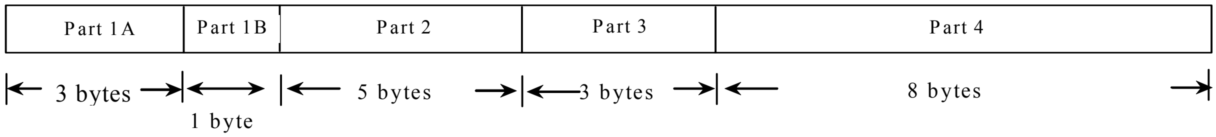


Figure 4-3: Overview of Segment Header Structure When All Parts Are Included

Table 4-3: Summary of Segment Header Functionality

Header Part	Length (Bytes)	Status	Description of Contents	Reference
Part 1A	3	Mandatory	Flags first and last segments of image; indicates which optional header parts are included; encodes information that typically changes from segment-to-segment	4.2.2
Part 1B	1	Mandatory for the last segment of image, not included otherwise.	Specifies number of ‘padding’ rows to be deleted after image reconstruction.	4.2.2
Part 2	5	Optional	Specifies limits on compressed bytes per segment and image quality	4.2.3
Part 3	3	Optional	Coding options including number of blocks per segment	4.2.4
Part 4	8	Optional	Image and compression parameters that must be fixed for an image.	4.2.5

NOTE – The mandatory first part of the header includes values of segment information that change from segment-to-segment. The optional second part of the header specifies limits on the number of compressed bytes in a segment and the limits on the fidelity with which DWT coefficients are encoded. This part might be included at the start of an image or application session, or at the beginning of each coded segment for variable output rate control. The optional third part of the header specifies information that is typically fixed for each image or application session, but is allowed to change with each segment. In a typical application, this part might be included at the beginning of each image, but not included for each segment. The fourth part of the header specifies parameters that must be fixed for an entire image.

4.2.2 SEGMENT HEADER PART 1

4.2.2.1 General

Segment Header Part 1 consists of two subparts:

a) Part 1A (3 bytes, mandatory for all segments),

b) Part 1B (1 byte, mandatory for the last segment, otherwise not used).

NOTE – The Segment Header Part 1 structure is illustrated in figure 4-4.

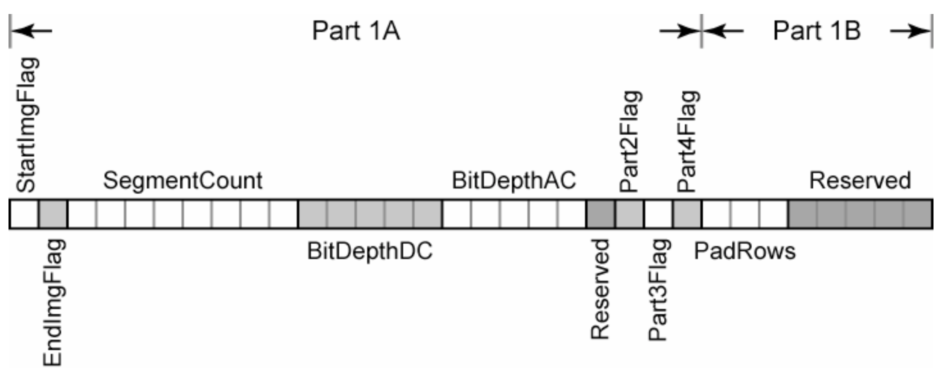


Figure 4-4: Segment Header Part 1 When Part 1B Is Included

4.2.2.2 Segment Header Part 1A

4.2.2.2.1 Bit 1 of Segment Header Part 1A shall contain the Start Image flag (StartImgFlag). The Start Image flag shall be used to indicate whether the segment corresponds to the start of an image:

– ‘1’ shall indicate that the segment is the first segment in the image;

– ‘0’ shall indicate that the segment is not the first segment.

4.2.2.2.2 Bit 2 of Segment Header Part 1A shall contain the End Image flag (EndImgFlag). The End Image flag shall be used to indicate whether the segment is the last segment in the image:

– ‘1’ shall indicate that the segment is the last segment in the image, in which case Segment Header Part 1B must be included;

– ‘0’ shall indicate that the segment is not the last segment, in which case Segment Header Part 1B shall not be included.

4.2.2.2.3 Bits 3-9 of Segment Header Part 1A shall provide the segment counter:

a) the eight-bit segment counter shall be encoded as an unsigned binary integer;

b) the segment counter shall equal zero for the first segment of an image and shall increment by one for each subsequent segment;

c) the segment counter shall wrap back to zero and resume incrementing after reaching a value of 255.

NOTE – This counter provides an index of segments, modulo 256, within the image. Absolute segment counting can be implemented using the CCSDS packet transport scheme as described in reference [2].

4.2.2.2.4 Bits 10-14 shall contain the BitDepthDC field. The value of the five-bit BitDepthDC field shall be encoded as an unsigned binary integer and shall equal the number of bits needed to represent DC coefficients for the segment in 2's complement representation.

4.2.2.2.5 Bits 15-19 shall contain the BitDepthAC field. The value of the five-bit BitDepthAC variable shall be encoded as an unsigned binary integer and shall equal the number of bits needed to represent the absolute value of AC coefficients in unsigned integer representation (see 4.4).

4.2.2.2.6 Bit 20 is reserved for future use by the CCSDS and shall be set to '0'.

4.2.2.2.7 Bit 21 of Segment Header Part 1A shall contain the Part 2 flag (Part2Flag) and shall indicate the presence or absence of optional Segment Header Part 2:

– '1' shall indicate that Segment Header Part 2 is present;

– '0' shall indicate that Segment Header Part 2 is absent.

4.2.2.2.8 Bit 22 of Segment Header Part 1A shall contain the Part 3 flag (Part3Flag) and shall indicate the presence or absence of optional Segment Header Part 3:

– '1' shall indicate that Segment Header Part 3 is present;

– '0' shall indicate that Segment Header Part 3 is absent.

4.2.2.2.9 Bit 23 of Segment Header Part 1A shall contain the Part 4 flag (Part4Flag) and shall indicate the presence or absence of optional Segment Header Part 4:

– '1' shall indicate that Segment Header Part 4 is present;

– '0' shall indicate that Segment Header Part 4 is absent.

4.2.2.3 Segment Header Part 1B

4.2.2.3.1 When EndImgFlag (4.2.2.2.2) in Segment Header Part 1A is set to '1', indicating that the segment is the last segment of the image, Segment Header Part 1B shall be included.

4.2.2.3.2 Bits 0-2 of Segment Header Part 1B shall contain the PadRows field. The value of the three-bit PadRows field shall be encoded as an unsigned binary integer and shall equal the number of 'padding' rows to be deleted (if any) after the inverse DWT is performed (see 3.2).

4.2.2.3.3 Bits 3-7 of Segment Header Part 1B are reserved for future use by the CCSDS and shall be set to '0'.

NOTE – Table 4-4 summarizes the contents of Segment Header Part 1.

Table 4-4: Contents of Segment Header Part 1

Part	Field	Width (bits)	Description	Details
Part 1A	StartImgFlag	1	Flags initial segment in an image	1: first segment in image 0: continuation segment in image
	EndImgFlag	1	Flags final segment in an image	1: last segment in image 0: otherwise
	SegmentCount	8	Segment counter value	segment count value (mod 256), encoded as an unsigned binary integer
	BitDepthDC	5	Number of bits needed to represent DC coefficients in 2's complement representation	value of *BitDepthDC* encoded as an unsigned binary integer.
	BitDepthAC	5	Number of bits needed to represent absolute value of AC coefficients in unsigned integer representation	value of *BitDepthAC* encoded as an unsigned binary integer.
	Reserved	1	Reserved for future use	0
	Part2Flag	1	Indicates presence of Part 2 header	1: Part 2 of header present 0: Part 2 of header absent
	Part3Flag	1	Indicates presence of Part 3 header	1: Part 3 of header present 0: Part 3 of header absent
	Part4Flag	1	Indicates presence of Part 4 header	1: Part 4 of header present 0: Part 4 of header absent
Part 1B	PadRows	3	Number of 'padding' rows to delete after inverse DWT	Value encoded as unsigned binary integer
	Reserved	5	Reserved for future use	00000

4.2.3 SEGMENT HEADER PART 2

4.2.3.1 General

If used, the optional Segment Header Part 2 shall specify output options that control the tradeoff between compressed data volume and reconstructed image quality for a segment.

NOTE – The Segment Header Part 2 structure is illustrated in figure 4-5.

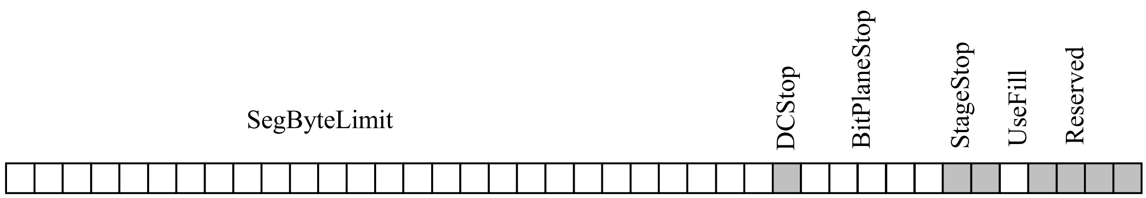


Figure 4-5: Segment Header Part 2

4.2.3.2 Contents of Segment Header Part 2

4.2.3.2.1 Bits 0-26 of Segment Header Part 2 shall contain the SegByteLimit field:

a) the value of the SegByteLimit field shall be encoded as an unsigned binary integer and shall equal the maximum number of bytes that can be used in a segment, including bytes used for the header;

NOTE – A SegByteLimit value less than 9 before the last segment and 10 for the last segment would not be permitted because of the combined number of bytes in the header Parts 1 and 2.

b) the value shall apply to the current segment and all subsequent segments until a new value of SegByteLimit is encoded in Segment Header Part 2 of a later segment header.

4.2.3.2.2 Bit 27 of Segment Header Part 2 shall contain the DCStop flag. The DCStop flag shall indicate the end of compressed output:

a) ‘1’ shall indicate that compressed output terminates once the quantized DC coefficient values (see 4.3.2) and additional bit planes (see 4.3.3) are encoded;

b) ‘0’ shall indicate that end of compressed output shall be determined by a bit plane index (BitPlaneStop) and a coding stage within the coding of that bit plane (StageStop).

NOTE – Coded segment data stops at the indicated point in the BPE coding process, or when the segment byte limit is reached, whichever comes first. Compressed data output always includes at least the coded quantized DC coefficients values and any additional DC bit planes (4.3.2 and 4.3.3) unless the SegByteLimit is too small to allow it.

4.2.3.2.3 Bits 28-32 of Segment Header Part 2 shall contain the BitPlaneStop field:

a) the 5-bit BitPlaneStop field shall be encoded as an unsigned binary integer and shall indicate the limit on coding of DWT coefficient bit planes;

b) when DCStop is set to ‘0’, the value of BitPlaneStop shall equal the index value of the bit plane in which coding stops;

c) when DCStop is set to ‘1’ the value of BitPlaneStop shall be ‘all zeros’.

4.2.3.2.4 Bits 33-34 of Segment Header Part 2 shall contain the StageStop field. The two-bit StageStop field shall indicate the stage at which the coding stops in the bit plane indicated by BitPlaneStop:

– ‘00’ shall indicate stage 1 (see 4.5.3);

– ‘01’ shall indicate stage 2 (see 4.5.3);

– ‘10’ shall indicate stage 3 (see 4.5.3);

– ‘11’ shall indicate stage 4 (see 4.5.4).

NOTE – Compression is limited entirely by the segment byte limit when DCStop is 0, BitPlaneStop is 0, and StageStop is set to 4. In this case, lossless compression is achieved for a segment when the integer DWT is used and the compressed data for the segment requires less than SegByteLimit bytes.

4.2.3.2.5 Bit 35 of Segment Header Part 2 shall contain the UseFill field, which shall indicate whether fill bits will be used to produce SegByteLimit bytes in each segment:

a) if the value of UseFill field is ‘1’, fill bits are used whenever needed to produce SegByteLimit bytes in each segment;

b) if the value of UseFill field is ‘0’:

   1) fill bits are not used to produce SegByteLimit bytes, and the number of compressed bytes in a segment may be less than SegByteLimit bytes;

   2) fill bits are used to fill to the next word boundary, when the stopping point (according to the specified values of DCStop, BitPlaneStop, StageStop) is reached before the byte limit specified in SegByteLimit;

NOTE – A word corresponds to the unit in which the compressor produces output, which may be a single byte, two bytes, or four bytes. For example, a compressor that produces 4 output bytes at a time might include as many as 31 fill bits in the last word of the compressed segment. The word length of the compressor is signaled by the field CodeWordLength in Segment Header Part 4 (see 4.2.5).

c) fill bits shall all be ‘all zeros’.

4.2.3.2.6 Bits 36-39 of Segment Header Part 2 are reserved for future use by the CCSDS and shall be set to ‘all zeros’.

NOTE – Table 4-5 summarizes the contents of Segment Header Part 2.

Table 4-5: Contents of Segment Header Part 2

<table>
<tr><th>Field</th><th>Width (bits)</th><th>Description</th><th>Details</th></tr>
<tr><td>SegByteLimit</td><td>27</td><td>Maximum number of compressed bytes in a segment</td><td>value (mod 2^{27}) encoded as an unsigned integer</td></tr>
<tr><td>DCStop</td><td>1</td><td>Indicates whether compressed output stops after coding of quantized DC coefficients (4.3).</td><td>1: Stop coding after coding quantized DC coefficient information (see 4.3.2) and additional DC bit planes (see 4.3.3)
0: Coding stop determined by BitPlaneStop and StageStop values</td></tr>
<tr><td>BitPlaneStop</td><td>5</td><td rowspan="2">Unused when DCStop = 1.

When DCStop = 0, indicates limit on coding of DWT coefficient bit planes. When BitPlaneStop = b and StageStop = s, compressed output stops once stage s of bit plane b bas been completed (see 4.5), unless coding stops earlier in the segment because of the segment byte limit (SegByteLimit).</td><td>Bit plane index value BitPlaneStop encoded as an unsigned integer.</td></tr>
<tr><td>StageStop</td><td>2</td><td>00: stage 1 (see 4.5.3)
01: stage 2 (see 4.5.3)
10: stage 3 (see 4.5.3)
11: stage 4 (see 4.5.4)</td></tr>
<tr><td>UseFill</td><td>1</td><td>Specifies whether fill bits will be used to produce SegByteLimit bytes in each segment.</td><td>1: fills bits used whenever needed to produce SegByteLimit bytes in each segment.
0: fills bits not allowed</td></tr>
<tr><td>Reserved</td><td>4</td><td>Reserved for future use</td><td>0000</td></tr>
</table>

4.2.4 SEGMENT HEADER PART 3

4.2.4.1 General

4.2.4.1.1 If used, Segment Header Part 3 shall specify the segment size in terms of number of *blocks*, *S*, defined in 4.1 and indicate whether the heuristic parameter selection approach (4.3.2) is used in the coding of quantized DC coefficients (see 4.3) and/or AC coefficient bit depths (see 4.4).

4.2.4.1.2 The information encoded in this part of the header is permitted to change with each segment.

NOTE – The Segment Header Part 3 structure is illustrated in figure 4-6.

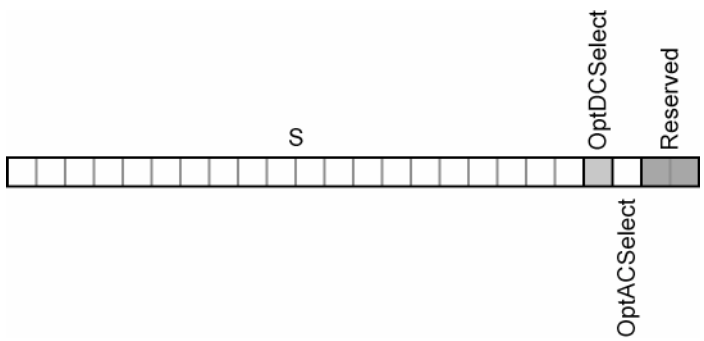


Figure 4-6: Segment Header Part 3

4.2.4.2 Segment Header Part 3 Contents

4.2.4.2.1 Bits 0-19 of Segment Header Part 3 shall contain the Segment Size field. The 20-bit Segment Size field shall be encoded as an unsigned binary integer and shall equal *S* mod 2^{20}; *S* shall be the number of blocks that make up the segment.

4.2.4.2.2 Bit 20 of Segment Header Part 3 shall contain the OptDCSelect field. The OptDCSelect field shall indicate the method used to select the value of the *k* parameter for coding quantized DC coefficient values (see 4.3.2.13):

– ‘1’ shall indicate optimum selection of *k*;

– ‘0’ shall indicate heuristic selection of *k*.

4.2.4.2.3 Bit 21 of Segment Header Part 3 shall contain the OptACSelect field. The OptACSelect field shall indicate the method used to select the value of the *k* parameter for coding AC coefficient bit depths (see 4.4):

– ‘1’ shall indicate optimum selection of *k*;

– ‘0’ shall indicate heuristic selection of *k*.

4.2.4.2.4 Bits 22-23 of Segment Header Part 3 are reserved for future use by the CCSDS and shall be set to ‘all zeros’.

NOTE – Table 4-6 summarizes the contents of Segment Header Part 3.

Table 4-6: Contents of Segment Header Part 3

Parameter	Width (bits)	Description	Details
S	20	segment size in blocks	Value encoded, mod 2^{20}, as an unsigned binary integer
OptDCSelect	1	Specifies whether optimum or heuristic method is used to select value of *k* parameter for coding quantized DC coefficient values (see 4.3.2)	1: optimum selection of *k* 0: heuristic selection of *k*
OptACSelect	1	Specifies whether optimum or heuristic method is used to select value of *k* parameter for coding *BitDepthAC* (see 4.4)	1: optimum selection of *k* 0: heuristic selection of *k*
Reserved	2	Reserved	00

4.2.5 SEGMENT HEADER PART 4

4.2.5.1 General

If used, optional Segment Header Part 4 shall provide information that may not change within an image, and that might often be fixed for an application session. This includes DWT type, optional user-defined weights for scaling the subbands, information about the input image, and coded word length.

NOTE – The Segment Header Part 4 structure is illustrated in figure 4-7.

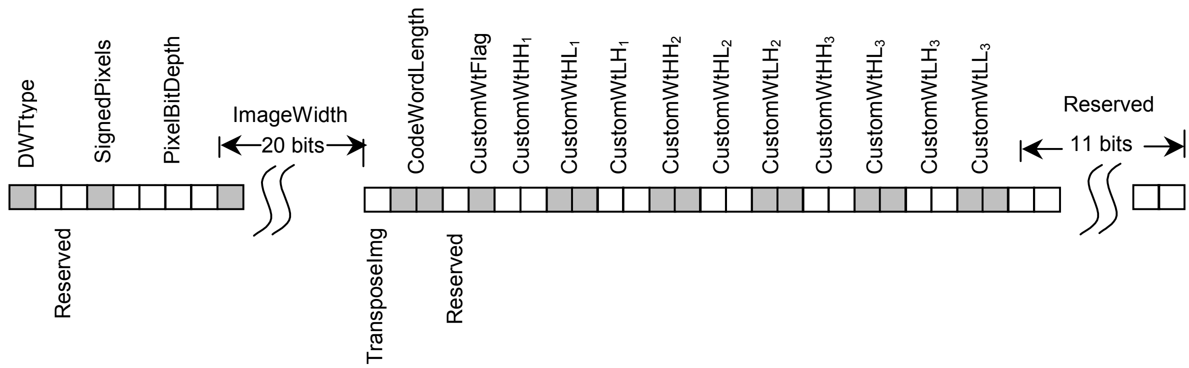


Figure 4-7: Segment Header Part 4

4.2.5.2 Segment Header Part 4 Contents

4.2.5.2.1 Bit 0 of Segment Header Part 4 shall contain the DWT Type field (DWTtype) and shall indicate the DWT type as follows:

- ‘0’ shall indicate real DWT;
- ‘1’ shall indicate integer DWT.

4.2.5.2.2 Bits 1-2 of Segment Header Part 4 are reserved for future use by the CCSDS and shall be set to ‘all zeros’.

4.2.5.2.3 Bit 3 of Segment Header Part 4 shall contain the Signed Pixels field (SignedPixels) and shall indicate whether input pixel values are signed or unsigned quantities:

- ‘0’ shall indicate unsigned;
- ‘1’ shall indicate signed.

4.2.5.2.4 Bits 4-7 of Segment Header Part 4 shall contain the Pixel Bit Depth field (PixelBitDepth). The four-bit Pixel Bit Depth field shall be encoded as an unsigned binary integer and shall equal the pixel bit depth mod 16.

4.2.5.2.5 Bits 8-27 of Segment Header Part 4 shall contain the Image Width field (ImageWidth). The 20-bit Image Width field shall be encoded as an unsigned binary integer and shall equal the width of the image in pixels mod 2^{20}.

4.2.5.2.6 Bit 28 of Segment Header Part 4 shall contain the Transpose Image field (TransposeImg) and shall indicate whether the entire image should be transposed after reconstruction:

- ‘0’ shall indicate that the image should not be transposed;
- ‘1’ shall indicate that the image should be transposed.

4.2.5.2.7 Bits 29-30 of Segment Header Part 4 shall contain the Code Word Length field (CodeWordLength) and shall indicate the length of the code word as follows:

– ‘00’: 8-bit word;

– ‘01’: 16-bit word;

– ‘10’: 24-bit word;

– ‘11’: 32-bit word.

4.2.5.2.8 Bit 31 of Segment Header Part 4 is reserved for future use by the CCSDS and shall be set to ‘0’.

4.2.5.2.9 Bit 32 of Segment Header Part 4 shall contain the Custom Weights flag (CustomWtFlag) and shall indicate whether standard (3.9) or user-defined weights are used:

– ‘0’ shall indicate that the subband weights defined in 3.9 are used;

– ‘1’ shall indicate that user-defined subband weights are used.

4.2.5.2.10 Bits 33-52 of Segment Header Part 4 shall provide 10 two-bit custom subband weight fields for each of the subbands HH_1–LL_3:

a) the custom subband weight fields shall be ordered as follows:

  – bits 33-34: HH_1,
  – bits 35-36: HL_1,
  – bits 37-38: LH_1,
  – bits 39-40: HH_2,
  – bits 41-42: HL_2,
  – bits 43-44: LH_2,
  – bits 45-46: HH_3,
  – bits 47-48: HL_3,
  – bits 49-50: LH_3,
  – bits 51-52: LL_3;

b) if CustomWtFlag is set to ‘1’, the values of the custom subband weight fields shall be set as follows:

  – ‘00’: weight = 2^0,
  – ‘01’: weight = 2^1,
  – ‘10’: weight = 2^2,
  – ‘11’: weight = 2^3;

c) if CustomWtFlag is set to ‘0’, the values of bits 33-52 shall be set to ‘all zeros’.

4.2.5.2.11 Bits 53-63 of Segment Header Part 2 are reserved for future use by the CCSDS and shall be set to ‘all zeros’.

NOTE – Table 4-7 summarizes the contents of Segment Header Part 4.

Table 4-7: Contents of Segment Header Part 4

Parameter	Width (bits)	Description	Details
DWTtype	1	Specifies DWT type	0: real DWT 1: integer DWT
Reserved	2	Reserved for future use	00
SignedPixels	1	Specifies whether input pixel values are signed or unsigned quantities	0: unsigned 1: signed
PixelBitDepth	4	Specifies the input pixel bit depth	Specifies values from 1 to 16 by mod(16) of 1, 2, 3, …, 16
ImageWidth	20	image width in pixels	Value encoded, mod 2^{20}, as an unsigned binary integer
TransposeImg	1	Indicates whether entire image should be transposed after reconstruction	0: do not transpose image 1: transpose image
CodeWordLength	2	Indicates the coded word length	00: 8-bit word 01: 16-bit word 10: 24-bit word 11: 32-bit word
Reserved	1	Reserved for future use	0
CustomWtFlag	1	Indicates if weights in 3.9 used or user defined	0: weights in 3.9 used 1: user-defined weights
CustomWtHH$_1$	2	Weight of HH$_1$ subband	00: weight = 2^0 01: weight = 2^1 10: weight = 2^2 11: weight = 2^3 (These fields are set to 00 when CustomWtFlag is 0)
CustomWtHL$_1$	2	Weight of HL$_1$ subband	
CustomWtLH$_1$	2	Weight of LH$_1$ subband	
CustomWtHH$_2$	2	Weight of HH$_2$ subband	
CustomWtHL$_2$	2	Weight of HL$_2$ subband	
CustomWtLH$_2$	2	Weight of LH$_2$ subband	
CustomWtHH$_3$	2	Weight of HH$_3$ subband	
CustomWtHL$_3$	2	Weight of HL$_3$ subband	
CustomWtLH$_3$	2	Weight of LH$_3$ subband	
CustomWtLL$_3$	2	Weight of LL$_3$ subband	
Reserved	11	Reserved for future	00000000000

4.3 INITIAL CODING OF DC COEFFICIENTS

4.3.1 GENERAL

4.3.1.1 The first step in coding DC coefficients in a segment shall be to encode a quantized representation of the DC coefficients using the predictive scheme described in this subsection. Bits providing further DC coefficient resolution shall be included as needed as part of the bit plane coding process of 4.5.

4.3.1.2 The amount of quantization of DC coefficients performed in this coding step shall be determined by the dynamic range of the AC and DC coefficients in a segment via the integer parameter q', which is defined according to table 4-8.

Table 4-8: DC Coefficient Quantization

DC and AC Dynamic Range	q' value	Remark
$BitDepthAC = 0$	$q' = 0$	All AC coefficients are zero.
$BitDepthDC \leq 3$	$q' = 0$	DC dynamic range is very small: no quantization is performed – all DC coefficient information is encoded in this step.
$BitDepthDC - (1+\lfloor BitDepthAC/2 \rfloor) \leq 1$ and $BitDepthDC > 3$	$q' = BitDepthDC - 3$	DC dynamic range is close to half the AC dynamic range: the 3 most significant bits of the DC coefficients are differentially coded at this step.
$BitDepthDC - (1+\lfloor BitDepthAC/2 \rfloor) > 10$ and BitDepthDC > 3	$q' = BitDepthDC - 10$	DC dynamic range is much higher than half the AC dynamic range: the 10 most significant bits of the DC coefficients are differentially coded at this step.
otherwise	$q' = 1+\lfloor BitDepthAC/2 \rfloor$	DC dynamic range is moderately higher than half the AC dynamic range: the DC coefficient bits exceeding half the AC dynamic range are differentially coded in this step.

4.3.1.3 DC quantization factor q in equation 15 is defined as:

$$q = \max(q', \mathrm{LS}(\mathrm{LL}_3)) \tag{15}$$

to take into account the DC bit planes that are necessarily all zeros as a result of the subband scaling operation.

NOTE – The value of q indicates the number of least-significant bits in each DC coefficient that are *not* encoded in the quantized DC coefficient values. The value of q is intended to be small enough that a significant amount of correlation in the DC coefficients can be exploited in the compression of these quantized coefficients. At the same time, the value of q is intended to be large enough to ensure that the BPE does not spend a large number of bits coding the DC coefficients to very high resolution before encoding any AC coefficient information.

4.3.1.4 For quantization in the initial DC coding phase, the DC quantization factor shall be compared to the dynamic range *BitDepthAC* of the AC coefficients of the segment, following the rules defined in table 4-8.

NOTE – The DC refinement bits required for coding DC coefficients with more precision are coded later, in stage 0 of Bit Plane Coding, as described in 4.5.

4.3.1.5 Next, given a sequence of DC coefficients $\{c_m | m=0,\ldots,S-1\}$ in a segment, the BPE shall compute quantized coefficients

$$c'_m = \lfloor c_m/2^q \rfloor \tag{16}$$

NOTE – When c_m is written in two's-complement representation, one can compute c'_m simply by performing q right bit shifts.

4.3.1.6 The quantized DC coefficients shall be encoded using the procedure described in 4.3.2, which effectively encodes the *BitDepthDC* - q most significant bits from each DC coefficient.

NOTE – When q >*BitDepthAC*, the q - *BitDepthAC* next most significant bits of each DC coefficient appear in the coded bitstream, as described in 4.3.3.

4.3.1.7 Remaining bits of DC coefficients appear in the coded bit stream as part of the bit plane coding procedure described in 4.5, except for the $\mathrm{LS}(\mathrm{LL}_3)$ least significant bits of each DC coefficient, which must be zero because of coefficient scaling, and are thus not encoded.

4.3.2 CODING QUANTIZED DC COEFFICIENTS

4.3.2.1 The number of bits needed to represent each quantized DC coefficient is given in equation 17 as:

$$N = \max\{BitDepthDC - q, 1\} \tag{17}$$

NOTE – The DC coefficient quantization rules in table 4-8 limit the value of N to be within 10, and thus coding options are defined only up to $N = 10$.

4.3.2.2 When N is 1, each quantized DC coefficient c'_m consists of a single bit. In this case, the coded quantized DC coefficients for a segment consist of these bits, concatenated together. Otherwise $N>1$ and the quantized DC coefficients in a segment, c'_m, shall be encoded using the procedure described below.

4.3.2.3 The first quantized DC coefficient for every sequence of S consecutive coefficients, referred to as a *reference sample*, shall be written to the encoded bitstream directly (i.e., without any further processing or encoding).

4.3.2.4 For the remaining S–1 DC coefficients, the difference between successive quantized coefficient values (taken in raster scan order) shall be encoded. Each difference value δ'_m in equation 18,

$$\delta'_m = c'_m - c'_{m-1} \tag{18}$$

shall be mapped to a non-negative integer δ_m according to the rules in equation 19:

$$\begin{aligned} \delta_m &= 2(\delta'_m), && \text{if } 0 \le \delta'_m \le \theta_m \\ \delta_m &= 2|\delta'_m| - 1, && \text{if } -\theta_m \le \delta'_m < 0 \\ \delta_m &= \theta_m + |\delta'_m|, && \text{otherwise} \end{aligned} \tag{19}$$

where $\theta_m = \min(c'_{m-1} - x_{min}, x_{max} - c'_{m-1})$, and $x_{min} = -2^{N-1}$, $x_{max} = 2^{N-1} - 1$.

4.3.2.5 The sequence of values δ_m shall then be entropy coded using a variable length code.

4.3.2.6 This procedure of prediction, mapping to nonnegative values, and coding operation shall follow the CCSDS Lossless Compression standard (reference [1]) with a slightly simplified entropy coder, described below, and different sequences to identify the code option selected for each block.

4.3.2.7 Within the segment of S DC coefficients, mapped quantized DC coefficients δ_m shall be further partitioned into groups referred to as *gaggles*, each gaggle containing up to 16 mapped quantized coefficients:

a) the first gaggle shall consist of 15 values of δ_m (the first quantized DC coefficient is coded directly as a reference sample);

b) the remaining gaggles shall each consist of 16 values of δ_m, with the possible exception of the last gaggle, which may have fewer samples;

c) if S is not a multiple of 16, then the last block shall contain J, which equals S mod 16, samples.

4.3.2.8 The values of δ_m in each gaggle shall be encoded using one of several code options ranging from uncoded (that is, each δ_m is encoded using the conventional N-bit unsigned binary integer representation) to encoded via one of several variable-length codes parameterized by a nonnegative integer k.

4.3.2.9 The code option selected for the block shall be indicated at the start of the coded gaggle using the appropriate *code option identifier* selected from table 4-9.

Table 4-9: Code Option Identifiers

Code Parameter *k*	***N*=2**	**2<*N* ≤ 4**	**4<*N* ≤ 8**	**8<*N* ≤ 10**
k = 0	0	00	000	0000
k = 1	–	01	001	0001
k = 2	–	10	010	0010
k = 3	–	–	011	0011
k = 4	–	–	100	0100
k = 5	–	–	101	0101
k = 6	–	–	110	0110
k = 7	–	–	–	0111
k = 8	–	–	–	1000
uncoded	1	11	111	1111
NOTE: ‘ – ’ indicates no applicable value.				

4.3.2.10 Otherwise, given code parameter k, the variable-length codeword for sample value δ_m has two parts. The first part shall consist of z zeros followed by a 1, where $z = \lfloor \delta_m / 2^k \rfloor$.

NOTES

1 z can be computed by performing k right bit-shifts of the binary representation of δ_m.

2 The structure of a coded gaggle is illustrated in figure 4-8; when the uncoded option is selected for a block, the coded gaggle has the structure indicated in figure 4-9.

4.3.2.11 The second part of the gaggle shall consist of the k least-significant bits of the binary representation of δ_m.

4.3.2.12 A coded gaggle shall consist of the code option identifier, then a reference sample (for the first gaggle in a segment), next the corresponding sequence of first part codewords for the samples in a gaggle, followed by the second part codewords.

4.3.2.13 For a given application, a user shall use one of the following methods for selecting the code option:

a) select the optimum value of k for each gaggle (i.e., select the value of k that minimizes the number of encoded bits in the coded gaggle) by explicitly computing the size of the encoded gaggle under each allowed value of k;

b) select the value of k using the following simplified (but in general suboptimal) heuristic procedure:

   Let Δ denote the sum of the δ_m values in a gaggle:

$$\Delta = \sum_m \delta_m \tag{20}$$

   The code option is selected depending on the value of the sum Δ according to rules in table 4-10.

Table 4-10: Code Option Selection Rules

Condition	Code Option Selected
$64 \cdot \Delta \geq 23 \cdot J \cdot 2^N$	uncoded
$207 \cdot J > 128 \cdot \Delta$	k=0
$J \cdot 2^{N-5} \leq 128 \cdot \Delta + 49 \cdot J$	k=N-2
otherwise	k is the largest nonnegative integer less than or equal to N-2 such that $J \cdot 2^{k+7} \leq 128 \cdot \Delta + 49 \cdot J$.

NOTE – Other methods of selecting the parameter k are not considered compliant with this Recommended Standard.

4.3.2.14 The method used to select the parameter k shall be indicated in Segment Header Part 4, when Segment Header Part 4 header is included.

NOTE – Since the value of the parameter k is explicitly encoded in the compressed data stream, it is not necessary for the decoder to know whether the optimum or the heuristic selection method was used.

4.3.2.15 The uncoded option shall be selected whenever it minimizes the number of encoded bits, even if another option gives the same number of bits.

4.3.2.16 When two or more code parameters minimize the number of encoded bits, the smallest code parameter option shall be selected.

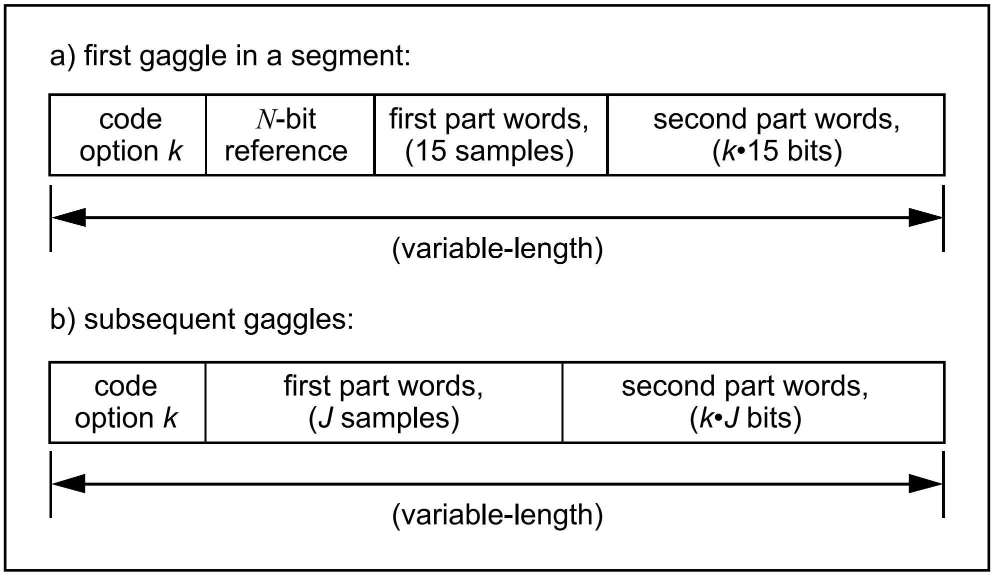


Figure 4-8: Coded Data Format for a Gaggle When a Coding Option Is Selected

NOTE – When the uncoded option is selected, the coded gaggle is fixed in length, consisting of the option ID field, optionally followed by an *N*-bit reference sample, and the *J* values of δ_m as shown in figure 4-9.

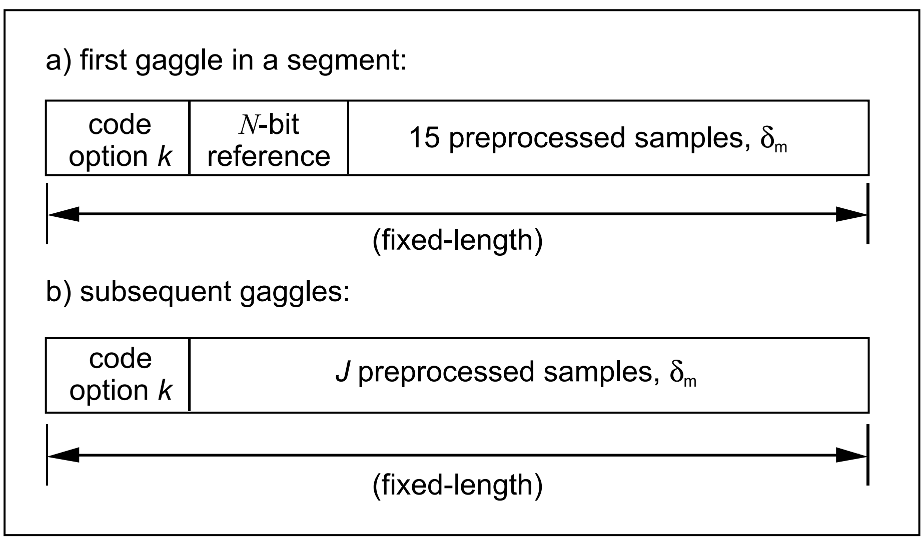


Figure 4-9: Coded Data Format for a Gaggle When No-Code Option is Selected

4.3.3 ADDITIONAL BIT PLANES OF DC COEFFICIENTS

NOTE – The coding of quantized DC coefficients described in 4.3.2 effectively encodes the first *BitDepthDC*-q bits of each DC coefficient.

4.3.3.1 When q >*BitDepthAC*, the next q-*BitDepthAC* bits of each DC coefficient shall appear in the coded bitstream.

4.3.3.2 The appropriate bits of each DC coefficient shall be concatenated for each bit plane; that is, the $(q\text{-}1)^{\text{th}}$ most-significant bit of each DC coefficient shall be followed by the $(q\text{-}2)^{\text{th}}$ most-significant bit, and so on, until the *BitDepthAC*$^{\text{th}}$ bit of each DC coefficient.

4.4 SPECIFYING THE AC BIT DEPTH IN EACH BLOCK

The first step in encoding AC coefficient magnitudes in a segment is to specify the sequence of $BitDepthAC\_Block_m$ values for the segment. These values shall be coded in one of three ways, depending on the value of *BitDepthAC* that is encoded in Part 1A of the segment header:

a) If *BitDepthAC* is zero, then the sequence of $BitDepthAC\_Block_m$ values shall all be zero and thus shall not be coded. In addition, no bit plane coding of the AC coefficients shall be performed since all of the AC coefficients in the segment must be zero.

b) If *BitDepthAC* is one, then each $BitDepthAC\_Block_m$ value shall be either zero or one, and thus each $BitDepthAC\_Block_m$ value shall be encoded using a single bit to indicate its value; i.e., the single-bit $BitDepthAC\_Block_m$ values shall be concatenated.

c) Otherwise, the sequence of $BitDepthAC\_Block_m$ values for the segment shall be coded using the same differencing and variable-length coding procedure specified for coding quantized DC values described in 4.3.2. However, a few parameters differ for the case of coding the $BitDepthAC\_Block_m$ values:

   1) N shall equal the number of bits required to represent the magnitude of *BitDepthAC* for the segment, as in equation 21:

   $$N = \lceil \log_2(1 + BitDepthAC) \rceil \tag{21}$$

   For code option identification, table 4-9 shall be used with this value of N.

   NOTE – For a 16-bit source image, no more than five bits are needed to specify the value of *BitDepthAC*, i.e., $N \leq 5$.

   2) Since the $BitDepthAC\_Block_m$ values are necessarily nonnegative, $x_{\min}$ and $x_{\max}$ shall be redefined as given in equation 22 and used to map successive differences to nonnegative integers:

   $$x_{\min} = 0, \quad x_{\max} = 2^N - 1 \tag{22}$$

NOTE – The coded bit string for the $BitDepthAC\_Block_m$ values follows the same data format illustrated in figures 4-8 and 4-9.

4.5 BIT PLANE CODING

4.5.1 OVERVIEW

Coding of a bit plane is performed in *stages* numbered 0-4. The coded bits produced at the stages for each block are interleaved, as illustrated in figure 4-10. Thus, a coded bit plane first consists of all the Stage 0 bits (if any) in the segment, then all of the coded Stage 1 bits in the segment, and so on, finishing with all of the encoded Stage 4 bits in the segment. This produces an embedded bit string with information from the highest bit plane of all S blocks in the first part of the output bit string followed by information from lower bit planes, and allows progressive decoding of the coded string. This improves image reconstruction quality when the coded bit sequence is truncated.

Stage 0 bits from each block in the segment (if any)
Coded Stage 1 from block 0, 1, …, S-1
Coded Stage 2 from block 0, 1, …, S-1
Coded Stage 3 from block 0, 1, …, S-1
Coded Stage 4 from block 0, 1, …, S-1

Figure 4-10: Coded Bit Plane Structure for a Segment

Note that when the index b of the bit plane being coded is larger than or equal to the AC bit depth of the block, then there is nothing to code for the block.

The layered coding stages in figure 4-10 inherently allow a controlled amount of DWT coefficient information to be encoded within a bit plane, permitting image quality control at sub-bit plane levels. The tradeoff between reconstructed image quality and compressed data volume for each segment is controlled by specifying the first four parameters in Part 2 Header (table 4-5). Quality level and data volume are both specified for each segment, and compressed output for a segment is produced until either the quality limit or the volume limit is reached. Users may choose to achieve fixed output rate by using fill bits. These feature are all described in 4.2.3.

Annex C contains a list of symbols used in the various coding stages specified in 4.5.2, 4.5.3, and 4.5.4.

4.5.2 OVERVIEW OF CODING STAGES

The coding stages for a block at bit plane b are described in the following paragraphs.

Stage 0 for a block consists of at most a single bit, which is simply the b^{th} most significant bit of the two's-complement representation of the DC coefficient. Note that whenever the bit plane index b satisfies $b \geq q$, this bit value is already known from the DC coefficient information already encoded, and in this case, stage 0 is empty. Stage 0 is also empty when scaling of the DC coefficient assures that the bit must be zero, i.e., when $b < \mathrm{LS}(\mathrm{LL}_3)$.

The remaining stages (1-4) encode AC coefficient bits. The stage in which bits from AC coefficients in a bit plane are coded depends on the *type* of the AC coefficient at the bit plane, which we now define. At bit plane b, the type of an AC coefficient x, denoted $t_b(x)$, has one of the following values:

– $t_b(x) = 0$ if $|x| < 2^b$, (x is not due for selection at this bit plane);

– $t_b(x) = 1$ if $2^b \le |x| < 2^{b+1}$, (x is due for selection at this bit plane);

– $t_b(x) = 2$ if $2^{b+1} \le |x|$, (x has already been selected at a previous bit plane);

– $t_b(x) = -1$ if $b < \mathrm{LS}(\Gamma)$, (x must be zero at this bit plane due to subband scaling).

Here, Γ denotes the subband containing x. Thus, during bit-plane encoding, each AC coefficient typically proceeds from type 0 to 1, to 2, to -1. For a set of coefficients Ψ, we define the type of the set, denoted $t_{\max}(\Psi)$, to be the maximum of the coefficient types in Ψ.

An AC coefficient x is said to be *selected* at bit plane b if $t_b(x) = 1$. I.e., the 'selection' of a coefficient marks the first bit plane where a non-zero magnitude bit is encoded for the coefficient. Note that $t_b(x) = 1$ if the bth most significant magnitude bit of x is the most significant '1' magnitude bit.

The type of a coefficient determines the stage when coding of a coefficient bit takes place. When an AC coefficient x is of type 0 or 1 (implying $t_{b+1}(x)=0$), the b^{th} most significant magnitude bit of x is coded in stages 1-3. Otherwise, the bit is included, uncompressed, in Stage 4 if x is of type 2, or not encoded at all when x is of type -1.

In **Stages 1-3** of BPE encoding bit plane b, the b^{th} magnitude bit of each AC coefficient x such that $t_{b+1}(x)=0$ is encoded. The b^{th} magnitude bits of the parent coefficients are coded in Stage 1, the children in Stage 2, and the grandchildren in Stage 3. Each of these stages also includes coded bits indicating the sign of each coefficient x for which $t_b(x)=1$. The coding in Stages 1-3 makes use of the family structure to group together AC coefficients for entropy coding.

The coding performed in stages 1-3 for a block consists of two parts. First, a sequence of variable length binary words are defined which completely describe the bits to be encoded in these stages, as discussed in 4.5.3.1. Next, a subset of these words are further entropy coded, as described in 4.5.3.3.

Stage 4 of coding consists of the b^{th} magnitude bit of each AC coefficient x with $t_b(x)=2$. These bits are included in the coded data stream uncompressed, in the order specified in 4.5.4.

4.5.3 CODING STAGES 1-3

4.5.3.1 AC Coefficient Words

NOTE – The bits encoded in stages 1-3 for a block can be determined by a sequence of words, as described below. Some of these words are further entropy coded, as described in 4.5.3.3.

4.5.3.1.1 In addition to the sets C_i, G_i, H_{ij}, defined in 4.1, P is defined as the list of parents in the block:

$$P = \{p_0, p_1, p_2\}.$$

4.5.3.1.2 The list of descendants in family i, denoted D_i, is defined as

$$D_i = \{C_i, G_i\}.$$

4.5.3.1.3 The list of descendants in a block, denoted B, is defined as

$$B = \{D_0, D_1, D_2\}.$$

NOTE – $\{A, B\}$ denotes the concatenation of the lists A and B.

4.5.3.1.4 A shorthand notation for certain binary words that describe information about bit plane b for a list of coefficients Ψ is defined as follows:

– let $types_b[\Psi]$ denote the binary word consisting of the b^{th} magnitude bit of each coefficient x in Ψ such that $t_b(x)$ equals 0 or 1;

– let $signs_b(\Psi)$ denote the binary word consisting of the sign bit of each coefficient x in Ψ such that $t_b(x) = 1$, with a sign bit of '1' for negative coefficients and '0' for nonnegative coefficients;

– given a list of type values $\Lambda=\{\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_l\}$, let $tword[\Lambda]$ denote the binary word consisting of the sequence of type values λ_i in Λ that are equal to 0 or 1.

NOTE – Any of these words can be null (i.e., have length zero).

4.5.3.1.5 The list P shall be ordered $P = \{p_0, p_1, p_2\}$, while the ordering on the lists C_i and H_{ij} shall be determined by the order in which their member coefficients' coordinates are listed in table 4-1.

4.5.3.1.6 The b^{th} magnitude bits for all AC coefficients that are type 0 at bit plane b+1 (i.e., not selected before the current bit plane) shall be communicated to the decoder by joining them to form binary words associated with each data type (parent, child, grandchild):

– $types_b[P]$;

– $types_b[C_i]$ for $i = 0, 1, 2$; and

– $types_b[H_{ij}]$ for $i = 0, 1, 2, j= 0, 1, 2, 3$.

4.5.3.1.7 At early bit planes, many sets of coefficients in a block tend to all be of type 0, and thus many of these words are initially all zeros. To effectively encode in this situation, the BPE shall make use of the following *transition* words to indicate when groups of coefficients at a lower depth are all Type 0:

– $tran_B = tword[\{t_{max}(B)\}]$;

– $tran_D = tword[\{t_{max}(D_0), t_{max}(D_1), t_{max}(D_2)\}]$;

– $tran_G$, which consists of concatenation of a single bit indicating the value of $t_{max}(G_i)$ for each family index $i = 0,1,2$ such that $t_{max}(G_i) \neq 2$ and $t_{max}(D_i) \neq 0$;

– $tran_{Hi} = tword[\{t_{max}(H_{i0}), t_{max}(H_{i1}), t_{max}(H_{i2}), t_{max}(H_{i3})\}]$ for $i = 0,1,2$.

4.5.3.1.8 At bit plane b, the BPE shall use the following sequence of words, generated in three stages, to update all of the AC coefficients in the block that were Type 0 at the previous bit plane:

a) **Stage 1 (parents)**:

   $types_b$[P], $signs_b$[P];

b) **Stage 2 (children)**:

   1) $tran_B$,

   2) $tran_D$, if $t_{max}(B) \neq 0$,

   3) $types_b$[Ci] and $signs_b$(C_i) for each i such that $t_{max}(D_i) \neq 0$;

   If $t_{max}(B) = 0$ then Stage 3 is unnecessary and shall be omitted;

c) **Stage 3 (grandchildren)**:

   1) $tran_G$,

   2) $tran_{Hi}$, for each i such that $t_{max}(G_i) \neq 0$,

   3) $types_b[Hij]$ and $signs_b[Hij]$ for each i such that $t_{max}(G_i) \neq 0$ and each j such that $t_{max}(H_{ij}) \neq 0$.

NOTE – All of the words generated in the above stages are variable length (including the null word).

4.5.3.1.9 Words $types_b[P]$, $types_b[Ci]$, $types_b[Hij]$, $tran_D$, $tran_G$, $tran_{Hi}$ shall be entropy coded, i.e., each shall be replaced with a corresponding variable-length codeword, whenever such a word has a length of at least 2 bits.

NOTE – The sign bit words are not coded further, because AC coefficients are generally positive and negative with about equal probability. The $tran_B$ word is always, at most, one bit in length and is never entropy coded.

4.5.3.2 Mapping Words to Symbols

4.5.3.2.1 The entropy coding procedure used to encode the words $types_b[P]$, $types_b[Ci]$, $types_b[Hij]$, $tran_D$, $tran_G$, $tran_{Hi}$ shall be accomplished through the use of variable-length codes given in 4.5.3.3. Words having a length of one bit, and sign-bit words, shall be included in the compressed data stream without further coding. All words of a given length shall be coded together, adaptively.

NOTE – Certain bit sequences cannot appear as values for certain words and this fact is taken into account in the entropy coding process. For example, $tran_D$ can never equal 000, because this condition would be inferred from the fact that $tran_B$=0. Table 4-11 summarizes the maximum word lengths and impossible values for each word that is entropy coded.

Table 4-11: Summary of Maximum Word Lengths and Impossible Word Values

Word	Maximum Length (bits)	Impossible Value
$types_b[P]$	3	–
$types_b[Ci]$	4	–
$types_b[Hij]$	4	0000
$tran_D$	3	000
$tran_G$	3	–
$tran_{Hi}$	4	0000

4.5.3.2.2 The process of variable-length coding of these words shall follow a two-step process:

– first, the word values shall be mapped to integer values referred to as *symbols*; then

– each integer shall be encoded using a variable-length binary codeword described in 4.5.3.3.

4.5.3.2.3 Under the mapping, two-bit, three-bit, and four-bit words shall be mapped to symbols using table 4-12, 4-13, or 4-14, respectively.

NOTE – The mapping process takes into account the fact that certain words can never be assigned certain bit sequences, as tabulated in table 4-11; this is reflected in tables 4-13 and 4-14.

Table 4-12: Integer Mapping for Two-Bit Words

Word	Symbol
00	0
01	2
10	1
11	3

Table 4-13: Integer Mapping for Three-Bit Words

Word	Symbol ($types_b[P]$, $types_b[Ci]$, $types_b[Hij]$, $tran_G$, $tran_{Hi}$)	Symbol ($tran_D$)
000	1	-
001	4	3
010	0	0
011	5	4
100	2	1
101	6	5
110	3	2
111	7	6

Table 4-14: Integer Mapping for Four-Bit Words

Word	Symbol ($types_b[Ci]$)	Symbol ($types_b[Hij]$, $tran_{Hi}$)
0000	10	-
0001	1	1
0010	3	3
0011	6	6
0100	2	2
0101	5	5
0110	9	9
0111	12	11
1000	0	0
1001	8	8
1010	7	7
1011	13	12
1100	4	4
1101	14	13
1110	11	10
1111	15	14

NOTE – The mappings are intended to produce symbol values in order of decreasing frequency. (I.e., the most frequently occurring word is mapped to symbol value 0, the next most frequent to 1, etc.) This makes effective coding possible through the coding procedure described in 4.5.3.3 because the codewords are arranged in order of increasing length. The mappings are based on statistics collected empirically for several test images.

4.5.3.3 Entropy Coding the Symbols

4.5.3.3.1 The symbols of 4.5.3.2 shall be encoded using the variable-length binary codes given in tables 4-15, 4-16, and 4-17.

4.5.3.3.2 As in the coding of DC coefficients, a segment of blocks shall be further partitioned into gaggles.

4.5.3.3.3 Each gaggle shall consist of 16 blocks, except for possibly the last gaggle in a segment, which shall contain S mod 16 blocks.

4.5.3.3.4 Within a gaggle,

– all two-bit words shall be encoded using one of the two variable-length code options given in table 4-15;

– all three-bit words shall be encoded using one of the three variable-length code options given in table 4-16;

– all four-bit words shall be encoded using one of the four variable-length code options given in table 4-17.

4.5.3.3.5 The variable-length codes used are permitted to change with each gaggle.

Table 4-15: Variable Length Code Options for Two-Bit Words

Input Symbol	Code Option 0	Code Option 1 (uncoded)
0	1	00
1	01	01
2	001	10
3	000	11

Table 4-16: Variable Length Code Options for Three-Bit Words

Input Symbol	Code Option 0	Code Option 1	Code Option 3 (uncoded)
0	1	10	000
1	01	11	001
2	001	010	010
3	00000	011	011
4	00001	0010	100
5	00010	0011	101
6	000110	0000	110
7	000111	0001	111

Table 4-17: Variable Length Code Options for Four-Bit Words

Input Symbol	Code Option 0	Code Option 1	Code Option 2	Code Option 3 (uncoded)
0	1	10	100	0000
1	01	11	101	0001
2	001	010	110	0010
3	0001	011	111	0011
4	0000000	0010	0100	0100
5	0000001	0011	0101	0101
6	0000010	000000	0110	0110
7	0000011	000001	0111	0111
8	00001000	000010	00100	1000
9	00001001	000011	00101	1001
10	00001010	000100	00110	1010
11	00001011	000101	00111	1011
12	00001100	0001100	00000	1100
13	00001101	0001101	00001	1101
14	00001110	0001110	00010	1110
15	00001111	0001111	00011	1111

4.5.3.3.6 The optimal code option (i.e., the code option that minimizes encoded length) shall be selected for each gaggle for each codeword size (i.e., two-bit, three-bit, and four-bit words). The code option selected shall be indicated using ID bits that take on the values given in table 4-18.

NOTE – At most five bits per gaggle per bit plane are needed to specify the code options.

4.5.3.3.7 The uncoded option shall be selected whenever it minimizes the number of encoded bits, even if another option gives the same number of bits. When two or more code parameters minimize the number of encoded bits, the smallest code parameter option shall be selected.

Table 4-18: Identifying the Variable Length Code Options

No. of Bits for Mapped Patterns	No. of ID Bits	ID and Associated Code Option
2	1	0: code option 0 1: uncoded
3	2	0: code option 0 1: code option 1 3: uncoded
4	2	0: code option 0 1: code option 1 2: code option 2 3: uncoded

4.5.3.3.8 The ID bits for specifying coding options for words of a given length shall be inserted immediately preceding the first appearance of a codeword for a given length within a gaggle. When no word of the given length occurs in a gaggle, no ID bits shall be present.

4.5.3.3.9 The coded word shall replace the appropriate uncoded word identified in 4.5.3.1.

4.5.4 STAGE 4 CODING

4.5.4.1 In stage 4 of coding, the b^{th} magnitude bit of each AC coefficient x with type $t_b(x)=2$ shall be included in the output bit stream.

4.5.4.2 For each block, the output bit string shall consist of the b^{th} magnitude bit of type 2 coefficients, in the following order:

– p_i, for each $i = 0,1$;

– members of C_i, for each $i = 0,1,2$;

– members of H_{ij}, for each $i = 0,1,2$, and each $j = 0,1,2,3$.

4.5.4.3 Members of the sets C_i and H_{ij} shall be processed in the order listed in table 4-1. No bits shall be coded in stage 4 for AC coefficients x not of type 2 ($t_b(x) \neq 2$).

4.5.4.4 The resulting strings for all blocks in the segment shall be concatenated to produce the entire stage 4 output string for the segment.

ANNEX A

GLOSSARY OF ACRONYMS AND TERMS

(Informative)

BPE	Bit-plane encoder: recommended processing algorithm used to compress wavelet coefficient data.
DWT	Discrete Wavelet Transform: recommended processing algorithm to transform image data to wavelet coefficient data.
Lifting scheme	Implementation scheme for DWT which performs transform in several subsequent 'lifting steps' requiring less operations than straightforward convolution scheme.
MSB	Most Significant Bit (of a word): left-most bit in figures: highest power of two in binary representation; first bit to be transmitted in serial output.
LL_n	2-d subband at n-th level of DWT; for n=1 obtained after sequentially applying low-pass 1-d DWT filter to horizontal and vertical lines of image; for n>1 obtained after sequentially applying low-pass 1-d DWT filter to horizontal and vertical lines of subband LL_{n-1}.
LH_n	2-d subband at n-th level of DWT; for n=1 obtained after sequentially applying low-pass 1-d DWT filter to horizontal, and high-pass 1-d DWT filter to vertical lines of image; for n>1 obtained after sequentially applying low-pass 1-d DWT filter to horizontal, and high-pass 1-d DWT filter to vertical lines of subband LL_{n-1}.
HL_n	2-d subband at n-th level of DWT; for n=1 obtained after sequentially applying high-pass 1-d DWT filter to horizontal, and low-pass 1-d DWT filter to vertical lines of image; for n>1 obtained after sequentially applying high-pass 1-d DWT filter to horizontal, and low-pass 1-d DWT filter to vertical lines of subband LL_{n-1}.
HH_n	2-d subband at n-th level of DWT; for n=1 obtained after sequentially applying high-pass 1-d DWT filter to horizontal and vertical lines of image; for n>1 obtained after sequentially applying high-pass 1-d DWT filter to horizontal and vertical lines of subband LL_{n-1}.
9/7 DWT	biorthogonal DWT with real-valued taps and recommended for lossy compression of image data.

9/7 integer DWT	non-linear, integer transform based on lifting scheme and recommended for lossy or lossless compression of image data.
weight factors	after transforming image data by means of the 9/7 integer DWT, the obtained wavelet coefficients need to be multiplied by these numbers before encoding with the BPE. One weight factor is defined for each subband.

ANNEX B

INFORMATIVE REFERENCES

(Informative)

[B1] *Procedures Manual for the Consultative Committee for Space Data Systems,* CCSDS A00.0-Y-9, Yellow Book. Issue 9. Washington, D.C.: November 2003.

[B2] *Image Data Compression*, Report concerning Space Data Systems Standards, CCSDS 123.0-G-1, Green Book. Issue 1 under preparation. Washington, D.C.: CCSDS, 2005.

[B3] ISO/IEC FCD15444-1, Information technology – JPEG2000 Image Coding System, Final Committee Draft Version 1.0, 2000.

[B4] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, July 1989.

ANNEX C

SYMBOLS USED IN CODING STAGES

(Informative)

Table C-1: Symbols Used in Coding Stages in Section 4

Category	Symbols	Meaning
wavelet coefficients	DC, c c' c'_{rec} c_{rec}	DC coefficient quantized DC coefficient reconstructed, quantized DC coefficient reconstructed DC coefficient
	AC, x p_i x'_{rec} x_{rec}	AC coefficient parent coefficient (i=0,1,2) reconstructed AC coefficient adjusted AC coefficient
	Γ	DWT subband
ordered sets of wavelet coefficients assigned to a block	Ψ B P D_i C_i G_i H_{ij}	variable set set of all descendent coefficients set of parent coefficients set of descendent coefficients (i=0,1,2) set of children coefficients (i=0,1,2) set of grandchildren coefficients (i=0,1,2) subset of set of grandchildren G_i (j=0,1,2)
integer valued functions	$t_b(x)$	value indicating type of coefficient x, with respect to current bit plane with index b
	$t_{\max}(\Psi)$	$\max\{t_b(x) \mid x \in \Psi\}$
binary-word valued functions	$\Psi \rightarrow funct[\Psi]$	general function description: An ordered coefficient set Ψ is mapped to a binary word *funct* by evaluating each member of the set $x \in \Psi$ however ignoring certain members x. The chronological order of the bits in the binary word *funct* corresponds to the ordering of the set Ψ.
	$types_b[\Psi]$	binary word whose bits are the types $t_b(x)$ of the coefficients $x \in \Psi$, ignoring coefficients with types other than $t_b(x)$=0 or $t_b(x)$=1
	$signs_b[\Psi]$	binary word whose bits are the signs of the coefficients $x \in \Psi$, ignoring coefficients whose types are not $t_b(x)$=1 Positive signs give 0, negative signs give 1.
	$tword[\{t_0,t_1,\ldots,t_l\}]$	binary words whose bits are the type values $\{t_0,t_1,\ldots,t_l\}$, ignoring types whose values are not 0 or 1

Table C-1: Symbols Used in Coding Stages in Section 4 (continued)

Category	Symbols	Meaning
binary words	$tran_{\mathrm{B}}$	transition word for set B
	$tran_{\mathrm{D}}$	transition word for collection of sets D_0, D_1, D_2
	$tran_{\mathrm{G}}$	transition word for collection of sets G_0, G_1, G_2
	$tran_{\mathrm{H}i}$	transition word for collection of sets H_{i0}, H_{i1}, H_{i2}, H_{i3} (i=0,1,2)